CERN-COMPUTING AND NETWORKS DIVISION
CN/95/5
March 1995

A Real-Time Application for the CS-2

Reiner Hauser & Iosif Legrand

Presented at the HPCN 95 Conference, Milan, May 1995

A Real-Time Application for the CS-2*

Reiner Hauser[1] and Iosif Legrand[2]

[1] CERN, 1211 Geneva 23, Switzerland
[2] DESY, 15738 Zeuthen, Germany

Abstract. Future high energy experiments planned at CERN for the next-generation collider LHC will generate amounts of data which are orders of magnitude larger than those produced by existing systems. A trigger system is used to reduce these data in real-time to an amount which can be stored on mass storage for later off-line analysis. Due to the high bandwidth and short latency constraints, the solution will most probably consist of a parallel processing implementation. We describe an implementation of the second-level trigger on a commercial multiprocessor system, the Meiko CS-2. Measurements show that such systems are already able to reach about 25 to 30 % of the required performance today, which makes them promising candidates for a complete solution in the next few years.

1 Introduction

The Large Hadron Collider (LHC) is a projected accelerator at CERN which will go into production in about 10 years. The high bunch crossing rate of 40 MHz and the resulting amount of data represent a challenge for data acquisition and triggers using currently existing computers and networks. The large amount of data is usually reduced by using several levels of triggers, which select 'interesting' events and discard uninteresting data.

The first level trigger will be implemented in highly parallelized custom hardware and is expected to reduce the event rate to 100 kHz and the total data bandwidth to about 100 GByte/sec. For the second-level trigger a number of possible architectures are under study, ranging from special-purpose systolic architectures [1][2] to farms of commercial workstation processors connected via a high-speed network switch like ATM, Fibre Channel or SCI. The goal is to take advantage of the expected technological progress during the next five to ten years in the field of processors and networks. Our goal is to investigate how closed parallel systems like the CS-2 can compete with custom-composed architectures.

The two main experiments proposed for LHC, Atlas [3] and CMS [4], follow slightly different approaches for implementing the second-level trigger system. The CMS solution uses a large general-purpose processor farm which incorporates both the second- and third-level trigger. All processors and data buffers are connected by a high-bandwidth switching network. Atlas follows the approach of dividing the trigger process into several clearly separated steps, where each

* This work was sponsored by ESPRIT project P7255 - GPMIMD II

step parallelizes and reduces the bandwidth further. This allows them e.g. to use different processor technologies and network switches in different parts of the trigger system, depending on the required bandwidth/latency/processing power etc. Fig. 1 shows the general structure of the Atlas trigger architecture.

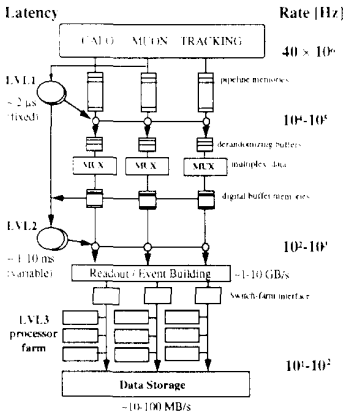

Fig. 1. Structure of Atlas trigger architecture

2 The Atlas Second-Level Trigger System

The current study has been carried out with the Atlas second level trigger architecture in mind. To avoid the transfer of all of the raw data from the detectors to the second level trigger, the first level trigger selects so called *Regions of Interest (RoI)*, which denote areas where interesting features were found. The algorithm for the second level trigger has been structured into several parts to take advantage of the inherent parallellism of the problem.

The so called *feature extraction* tasks work on a single RoI of a given subdetector. In our implementation all the features from different subdetectors belonging to a given RoI are collected into the *RoI task*. Finally, the so called *global task* collects the data from all the RoIs to compute the final decision.

The *feature extraction* tasks receive the data from the detector for one RoI. This is typically in the order of a few kBytes/event. The computed features which are sent to the RoI task are usually only a few bytes. In the following we assume that packets are smaller than 64 bytes/event. The complete raw data are stored in buffers until the second-level trigger has made the decision whether they should be transfered to the third-level trigger or discarded. The size required for these buffers depends obviously on the latency of the whole trigger system, i.e. the time it takes to make the final decision.

Fig. 2 shows the structure of the corresponding program as a graph, where the nodes represent computation task and the edges the data flow between the different tasks.

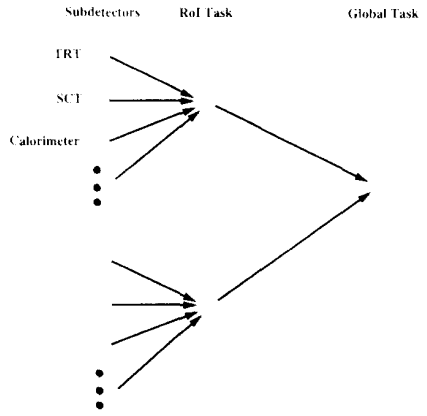

Fig. 2. Structure of second-level trigger program

3 The CS-2 at CERN

The Meiko CS-2 is a distributed memory machine, using SPARC CPUs and the Solaris operating system on each processing node. The nodes are connected by a high-speed, low-latency network. The system installed at CERN as part of the ESPRIT project GPMIMD-II consists of 32 nodes, mostly with 32 MByte of memory each and a processor clock speed of 40 MHz.

The distinguishing feature of the CS-2 is its internal network switch. Transfers can be initiated by a normal user program without requiring any system call. The actual data transfer is handled by a communication processor [5], which allows the main CPU to continue with other operations.

The communication facilities can be used from high-level message passing libraries like MPI [6], but there is also a low-level interface, called the Elan Widget library [7]. It has an application interface built on logical *channels* between different processors. There is also a direct memory access (DMA) interface, which allows e.g. data transfers to a given address on a remote processing node without intervention on the receiver side.

Fig. 3 shows the results of some bandwidth measurements for DMA and channel communication. For large packets, transfer rates of more than 40 MByte/sec can be achieved. However, for small packets (less than 200 Bytes) the transfer time is usually dominated by the startup latency of the network which is in the order of 10 μs on the lowest level. When using the channel interface the startup latency increases to about 25 μs.

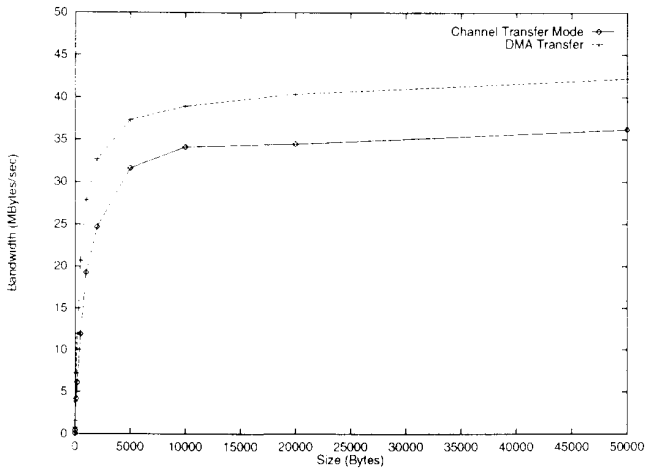

Fig. 3. Bandwidth measurements on the CS-2

4 Implementation

We have implemented the global decision part of the second level trigger algorithm on the Meiko CS-2 multiprocessor system. In our setup each feature extraction, RoI and global task is mapped to a single processor. The feature extraction algorithms are not actually implemented in the nodes; instead the programs generate random data and feed it into the rest of the system as fast as possible. Since the communication channels include a flow control mechanism, we have an easy way of measuring the event rate with which the whole system can keep up.

The RoI task receives packets and puts them into a list until all the data for a given RoI have been collected. Buffers are never copied when they are in memory. Only pointers to them are passed around. When all expected packets have arrived, the actual procedure for the RoI algorithm is called. This routine typically executes in less than 10 μs. The results are then sent to the global task.

The global task is structured in a similar way. A central loop collects packets from different RoIs, until all the data for a given event have been received. The execution time for the global task is dependent on the number of RoIs for the current event. It takes typically about 30 μs to perform its task.

The communication library of the CS-2 allows for asynchronous transfers. All parts of the program take advantage of this property to overlap their communication with their computation. They always initiate a request for sending or receiving a packet, and then use polling to test for completion if necessary.

The communication part of the program has been implemented using the two different methods described above: first with the virtual channel based method, then with direct DMA transfers. The channel implementation is much more natural for the programmer. Buffers can be allocated as needed and just sent to

the receiver node without caring about the buffers on the destination side. The receiver may also choose an arbitrary buffer and issue an asynchronous receive request, then continue with some other task.

The disadvantage of this scheme is that is requires several packet exchanges between the communication processors to get the destination address and then transfer the data to that buffer. Furthermore the communication library requires that the programmer polls each outstanding transmission separately and does one final function call to complete the transfer.

The DMA implementation uses a different approach: a large number of buffers on the receiver side are preallocated, then a list of the addresses of these buffers is sent to the sender node. The sender is allowed to transfer data to the receive buffers without further permission and without waiting for an acknowledge. Only after it has used all the buffers, does it have to wait for a new list of addresses from the receiver. This reduces the overhead for a single packet transfer. The overhead for the necessary synchronisation is amortized over the large number of buffers sent without further intervention on the receiver side. On the receiver side the processor just checks if anything new has arrived in the buffers (indicated by a special flag in the packets).

5 Results

The system was run with a varying number of parameters for the number of subdetectors and the number of RoIs. These two parameters determine the total number of processors needed (see Fig. 2). The maximum numbers were determined by the current setup of the CS-2. Each basic task was assigned to a single processor node. The measurements were done by sending one million events into the system and computing the mean value for one event. From this a event rate can be calculated for the given setup.

The number of subdetectors varied from two to four, and the number of RoIs from two to four. For the channel implementation, event rates vary from about 20 to 40 kHz, depending on the parameters and the communication method used. As expected, the achievable rate decreases with an increasing number of subdetectors/RoIs. Table 1 shows the results.

As one can see the DMA implementation is usually faster than the channel implementation. A more interesting point is that the speedup of the DMA implementation over the channel version increases when the system as a whole is getting larger. This indicates that the overhead of the channel library may not be neglected and that the polling of a large number of incoming channels may take up some considerable time. There are two columns for the DMA implementation, one with 512 buffers and one with 2048. As one can see, the performance of the latter is usually better for small systems, while it decreases for large systems. This is due to the increased communication which is necessary to send the addresses for buffer pool: for 2048 buffers about 8 kByte of data has to be transfered to each sending processor.

RoIs	Subdetectors	DMA (512 bufs)	DMA (2048 bufs)	Channel	Processors used
2	2	34.8	41.2	33.7	7
2	3	29.7	32.6	26.9	9
2	4	27.9	24.9	20.9	11
3	2	31.5	34.5	27.5	10
3	3	27.6	30.2	22.9	13
3	4	25.2	24.2	17.8	16
4	2	30.2	33.4	25.2	13
4	3	27.7	23.5	18.3	17
4	4	21.1	17.7	13.8	21

Table 1. Timing measurements: achieved event rate in kHz

6 Discussion

We have shown that is possible to implement an algorithm for the global decision part of the LHC second level trigger on a commercial multiprocessor system, the Meiko CS-2, achieving a throughput which is only a factor three to five below the required rate.

A new version of the communication processor, the ELAN-2 will be available at the end of this year. The latency for DMA transfers will be reduced to less than 3 μs. In the near future the CERN machine will be upgraded to 100 MHz processor nodes, thereby doubling the processing performance. These two factors together should increase the performance of the application by at least a factor of two, bringing the system quite close to the required limit.

References

1. J. Badier et al., IEEE Trans. Nucl. Sci. **40**, 1993.
2. D. Belosloudtsev et al., to appear in NIM, Feb 1995.
3. Atlas - Technical Proposal for a General Purpose Experiment at the Large Hadron Collider at CERN, CERN/LHCC/94-43, 1994.
4. CMS The Compact Muon Solenoid - Technical Proposal, CERN/LHCC/94-38, 1994.
5. Communications Processor Overview, Meiko Computing Surface Documentation, 1993.
6. MPI: A Message-Passing Interface Standard, Version 1.0, 1994.
7. Elan Widget Library, Meiko Computing Surface Documentation, 1993.